CLAIMS

We claim:

1. A multi-tasking operating system for managing simultaneous access to scarce or serially re-usable resources by multiple process threads, comprising:

at least one resource;

a plurality of threads; and

a stationary queue for allocating access to said resource amonst said threads.

2. A multi-tasking operating system stationary queue for managing simultaneous access to scarce or serially re-usable resources by multiple process threads, the stationary queue comprising:

a sleep code routine for generating a unique block identifier when a process thread temporarily cannot gain access to said resource and must be suspended; and

a wakeup code routine for generating a unique run identifier when a next thread in line is to be re-animated and granted access to said resource.

3. The system of claim 2, further comprising:

a wait counter for counting the cumulative number of threads that have been temporarily denied the resource;

a satisfied counter for counting the cumulative number of threads that have been denied access and subsequently granted access to said resource;

said sleep code routine being responsive to said wait counter for generating said run identifier; and

said wakeup code routine being responsive to said satisfied counter for generating said run identifier.

4. A method for managing simultaneous access to scarce or serially re-usable resources by multiple process threads, comprising the steps of:

responsive to a request for a resource which is not available, creating a block identifier based on the number of threads temporarily denied the resource; and

blocking the thread using said block identifier.

5. A method for managing simultaneous access to scarce or serially re-usable resources by multiple process threads, comprising the steps of:

responsive to a resource becoming available, creating a run identifier based on the number of threads that have been first forced to wait and have been subsequently satisfied; and

running the thread using said run identifier.

6. A method for managing simultaneous access to scarce or serially re-usable resources by multiple process threads, comprising the steps of:

responsive to a request for a resource which is not available,

creating a block identifier based on the number of threads temporarily denied the resource; and

blocking the thread using said block identifier; and

responsive to a resource becoming available,

creating a run identifier based on the number of threads that have been first forced to wait and have been subsequently satisfied; and

running the thread using said run identifier.

7. A memory device for storing signals for controlling the operation of a computer to manage simultaneous access to scarce or serially re-usable resources by multiple process threads, according to the steps of

responsive to a request for a resource which is not available,

creating a block identifier based on the number of threads temporarily denied the resource; and

blocking the thread using said block identifier; and

responsive to a resource becoming available,

creating a run identifier based on the number of threads that have been first forced to wait and have been subsequently satisfied; and

running the thread using said run identifier.

8. A memory device for storing signals to structure the components of a digital computer to form a stationary queue for managing simultaneous access to scarce or serially re-usable resources by multiple process threads, comprising:

a sleep code routine for generating a unique block identifier when a process thread temporarily cannot gain access to said resource and must be suspended;

a wakeup code routine for generating a unique run identifier when a next thread in line is to be re-animated and granted access to said resource;

a wait counter for counting the cumulative number of threads that have been temporarily denied the resource;

a satisfied counter for counting the cumulative number of threads that have been denied access and subsequently granted access to said resource;

said sleep code routine being responsive to said wait counter for generating said run identifier; and

said wakeup code routine being responsive to said satisfied counter for generating said run identifier.